XpertVR Conversational AI Report

Rahul Vaghasia

Theresa Merin

Veena Nagapurkar

Vrindha

AI Project – XpertVR Conversational AI

Index

1. Problem Statement

Conversational AI capabilities can be developed for the NPCs within a game that can have human-like interaction with player. This can be done using training a machine learning model on large dataset of Natural Language. This a complex process and requires a huge amount of dataset and computational resources in order to develop just workable results. The workaround this problem is the pre-trained language model that are made available by OpenAI and Google. The GPT-3 model by OpenAI is the most advanced transformer model and BERT made by Google is also a transformer model.

2. Objective

2.1 General Objective

The general objective of the project is to work on two different pathways to achieve AI conversations in NPC within a game. One is to fine-tune the GPT-3 model to incorporate in NPC to provide text-generation as a response to the question asked by the player playing the game. The other pathway is to experiment with Inworld – an online platform where the AI NPC characters can be developed with specific personality traits, behaviour, emotions, background story, etc. Both the pathways are cost run.

2.2 Specific Objectives

The specific objectives of this project are as follows:

- To investigate and evaluate different natural language generation models, such as GPT, for use in the AI.
- To integrate the AI into XpertVR's VR simulation application using natural language generation models.
- To enable XpertVR to build more realistic immersive experiences for their users through more realistic interactions with nonplayer characters in a variety of training and teaching simulations.
- Testing the Inworld platform in Unity engine and compare the performance with the GPT-3.

3. Methodology

3.1 Project Planning

1) Requirement analysis
    - About the project

- Project phases
- SDLC
- Technology and tools being used.
- What type of testing will be done?
- Type of report that will be used.

2) Research
    - About GPT-3 model
    - API
    - Dataset
    - Plugins
3) Module 1:
    - ➢ Development 1
        - Phase 1: developing a smaller dataset.
    - ➢ Testing 1
        - Phase 1: testing the smaller dataset to ensure all the connections are intact.
    - ➢ Release 1
        - Deploy it to the client and ensure they're satisfied and get feedback for any improvements to deploy it to the next module.
4) Module 2:
    - ➢ Development 2
        - Phase 2: Develop a larger dataset and the improvements suggested.
    - ➢ Testing 2
        - Phase 2: Testing the larger dataset to ensure all the connections are intact.
    - ➢ Release 2
        - Deploy it to the client and ensure they're satisfied and get feedback for any improvements to deploy it to the next module.
5) Module 3:
    - ➢ Development 3
        - Phase 3: developing voice according to the emotion.
    - ➢ Testing 3

        - Phase 3: Checking whether the emotions are working for the voice.
    - ➢ Release 3
        - Deploy it to the client and ensure they're satisfied and get feedback for any improvements to deploy it to the next module.

6) Deployment
7) Maintenance

3.2 System Analysis

The goal of this project is to use GPT-3 to provide conversational capabilities to nonplayable characters (NPCs) in games. This will be accomplished by providing character descriptions of the NPCs, along with the questions asked by the player as input to GPT2. The model will then generate contextual answers based on the information provided in the character descriptions.

To achieve this goal, the following system components will be required:

1. GPT-3 model: the GPT-3 model will be trained on a large dataset of character descriptions and questions/answers to learn the relationship between the two.

2. Flask API: the Flask API will provide an interface for the game to communicate with the GPT-3 model. The game will provide the character description and question as input to the API, which will then call the GPT-3 model and return the generated answer.

3. Unity Engine integration: the game itself will need to be modified to provide character descriptions and send questions to the Flask API. The game will also need to display the generated answers to the player.

To properly analyze the system, the following factors will need to be considered:

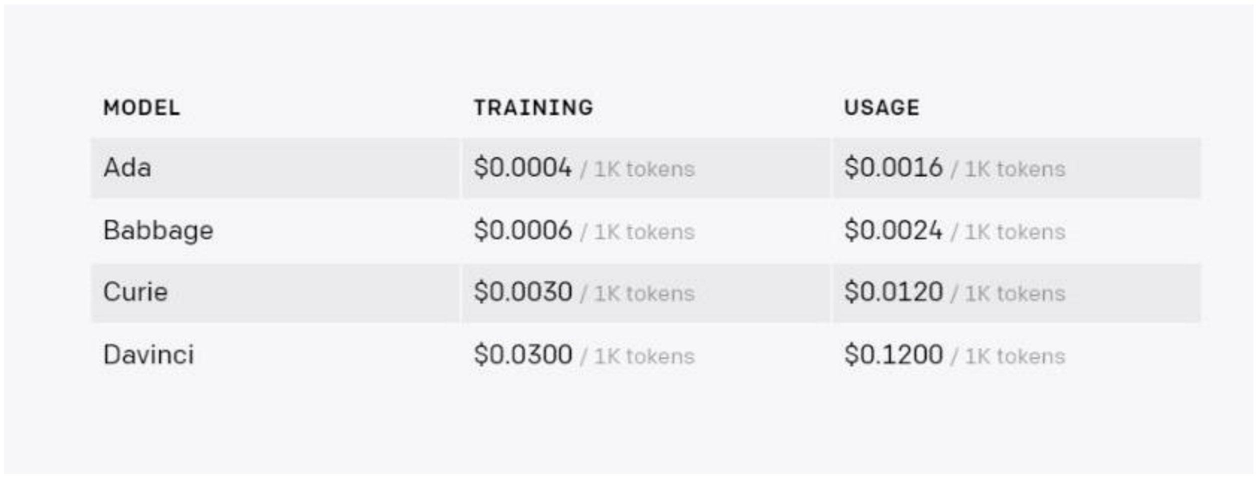
1. Data quality: the quality of the training data will have a significant impact on the performance of the GPT-3 model. In order to generate accurate and relevant answers, the character descriptions and questions/answers in the training data must be well-written and diverse.

2. Model performance: the performance of the GPT-3 model will need to be evaluated to ensure that it is able to generate accurate and relevant answers. This can be done by comparing the model's answers to a set of humangenerated answers for the same character descriptions and questions.

3. Integration with Unity Engine: the Flask API and GPT-3 model will need to be integrated seamlessly with the game to ensure a smooth and intuitive experience for the player. This will require careful planning and coordination between the game developers and the team implementing the GPT-3 model. Overall, the use of GPT-3 to provide conversational capabilities to NPCs in games has the potential to greatly improve the realism and immersion of the game experience. By carefully considering the quality of the training data, the performance of the

GPT-3 model, and the integration with the game, this system has the potential to deliver a highly engaging and realistic conversational experience for players.

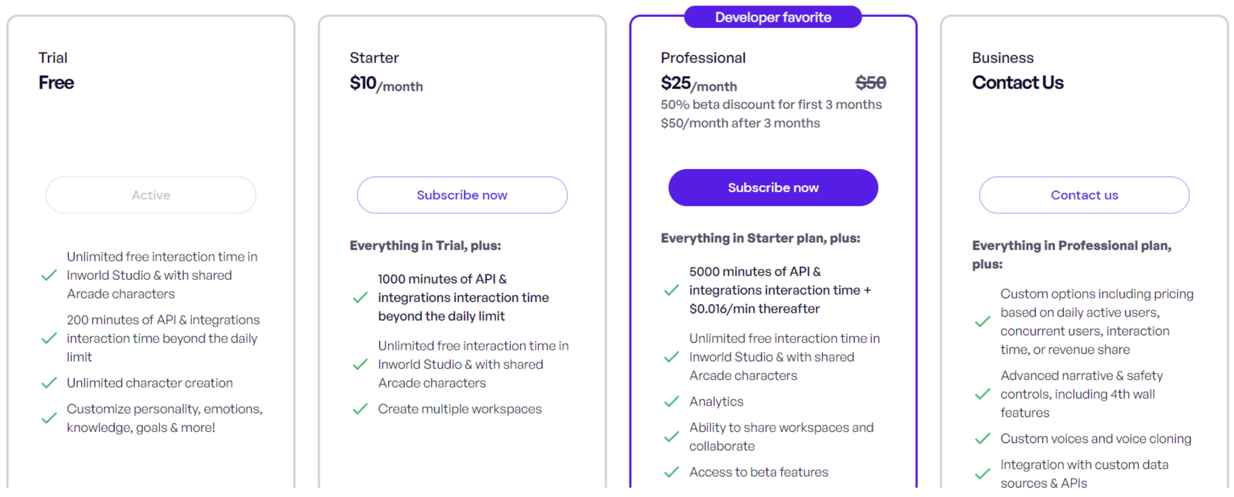
OpenAI GPT-3 access is subscription based. The cost table for the GPT-3 is:



MODEL	TRAINING	USAGE
Ada	$0.0004 / 1K tokens	$0.0016 / 1K tokens
Babbage	$0.0006 / 1K tokens	$0.0024 / 1K tokens
Curie	$0.0030 / 1K tokens	$0.0120 / 1K tokens
Davinci	$0.0300 / 1K tokens	$0.1200 / 1K tokens

We are focusing on using Davinci model. While training the model on dataset the cost will be 0.03 USD per 1000 tokens of text. Usage will cost 0.12 USD per 1000 tokens of text. Based on the information available on GPT-3 website, the first 18 dollars are free to use.

Inworld platform is also a subscription-based platform. The cost table is given below:



Trial
Free

Active

- ✓ Unlimited free interaction time in Inworld Studio & with shared Arcade characters
- ✓ 200 minutes of API & integrations interaction time beyond the daily limit
- ✓ Unlimited character creation
- ✓ Customize personality, emotions, knowledge, goals & more!

Starter
$10/month

Subscribe now

Everything in Trial, plus:

- ✓ 1000 minutes of API & integrations interaction time beyond the daily limit
- ✓ Unlimited free interaction time in Inworld Studio & with shared Arcade characters
- ✓ Create multiple workspaces

Developer favorite

Professional
$25/month ~~$50~~
50% beta discount for first 3 months
$50/month after 3 months

Subscribe now

Everything in Starter plan, plus:

- ✓ 5000 minutes of API & integrations interaction time + $0.016/min thereafter
- ✓ Unlimited free interaction time in Inworld Studio & with shared Arcade characters
- ✓ Analytics
- ✓ Ability to share workspaces and collaborate
- ✓ Access to beta features

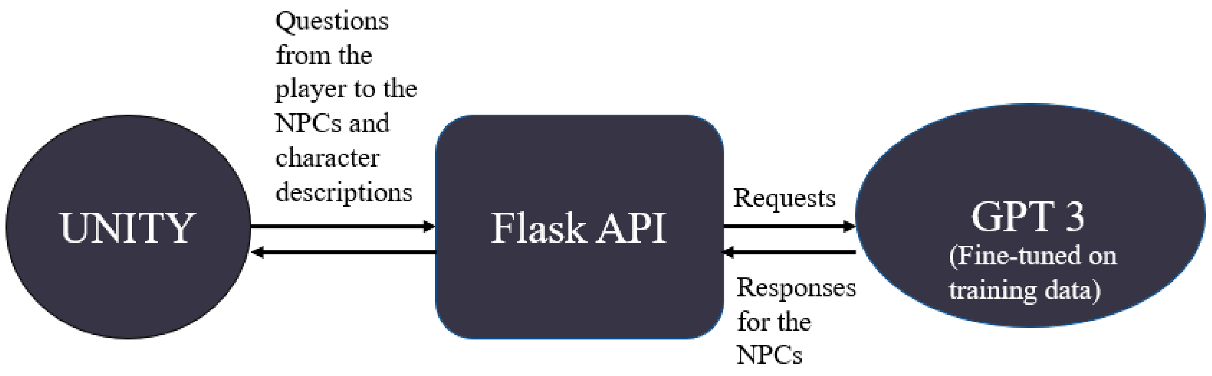
Business
Contact Us

Contact us

Everything in Professional plan, plus:

- ✓ Custom options including pricing based on daily active users, concurrent users, interaction time, or revenue share
- ✓ Advanced narrative & safety controls, including 4th wall features
- ✓ Custom voices and voice cloning
- ✓ Integration with custom data sources & APIs

3.3 System Design


Questions from the player to the NPCs and character descriptions
UNITY
Flask API
Requests
Responses for the NPCs
GPT 3
(Fine-tuned on training data)

The system will be developed as a Flask API, and the input to the API will be the character description and the query. The produced answer will then be returned to the game by the API after a call to the GPT-3 model. The response will be shown to the player via the game.

The main parts of the system are:

1. GPT-3
2. Flask API
3. OpenAI API
4. Interface Plugins

The interface between Unity and fine-tuned model is done through FlaskAPI. There is another way to do this as well. That is by adding OpenAI API functionality within Unity which is currently not approachable option given the time constraints.

The interface would work in such a way that the Unity would send a POST request with a question to Flask API which would then forward the request to OpenAI API to get the response back from fine-tuned model. The response from fined-tuned model is then given back to Unity which is then displayed in the game.

For the first time when the request is sent, it needs to be provided with the information about the NPC to which the conversation is going to take place. This can be done by having the background information about the NPCs on local database from where the information can be taken and forwarded along with the question from Unity to the fined-tuned model.

In order to facilitate a systematic flow we need to create a system that would store the history about the conversation with NPC so that would have the memory of this conversation.

The system will be implemented as a Flask API, where the character description and question will be provided as input to the API. The API will then call the GPT-3 model and return the generated answer to the game. The game will display the answer to the player.

The following steps will be taken to implement the system:

1. Train the GPT-3 model: the GPT-3 model will be trained on a large dataset of character descriptions and questions/answers to learn the relationship between the two. This will be done using a combination of supervised and unsupervised learning techniques.

2. Implement the Flask API: the Flask API will be implemented using Python and Flask. The API will provide an interface for the game to communicate with the GPT-3 model and will handle authentication and encryption of data.

3. Integrate the API with the game: the game will be modified to provide character descriptions and send questions to the Flask API. The game will also need to display the generated answers to the player.

4. Test and evaluate the system: the system will be tested and evaluated to ensure that it is able to generate accurate and relevant answers in a variety of simulated scenarios. This will involve comparing the model's answers to a set of human generated answers for the same character descriptions and questions.

Once the system has been implemented and tested, it will be deployed in a production environment where it can be used by games to provide conversational capabilities to NPCs. The system will be monitored and maintained to ensure its continued performance and reliability.

Overall, the proposed system implementation is well-suited to providing conversational capabilities to NPCs in games, and is scalable, secure, and reliable.

3.4 System Testing

A thorough testing technique will be used to make sure the system can produce accurate and pertinent answers. The following steps will be required for this:

1. Unit testing: To make sure that each component of the system is working properly, such as the GPT-3 model and the Flask API, each component will be tested individually.

2. Integration testing: the various system parts will be examined collectively to check that they can function as a unit.

3. System testing: To make sure the system can produce correct and pertinent answers in a range of simulated circumstances, the complete system—including the game integration—will be tested.

4. User acceptance testing: A group of users will test the system to see if it is simple to use and intuitive.

The system's performance will be assessed using the following standards:

1. Accuracy: By contrasting the system's responses with a collection of human-generated responses to the same character descriptions and questions, the system's capacity to provide accurate and pertinent replies will be assessed.

2. Fluency: The system's responses must be well-written and grammatically sound.

3. Relevance: The system's responses must be pertinent to the query and character description that were given.

4. Response time should be under one second, and the system should be able to deliver results quickly.

The system will be deemed prepared for deployment in a production setting if it can satisfy these requirements. To verify that the system can produce correct and pertinent answers in a number of simulated circumstances, a thorough testing technique will be used. To ensure the system's functionality and dependability, both the system's individual parts and the system as a whole will be tested.

3.5 Acceptance, Installation and Deployment

Acceptance Standards

- The Flask API should be able to accept input in the form of both character descriptions and questions.
- With the supplied input, the Flask API ought to be able to call the GPT-3 model and deliver the contextual response which should be smoothly.
- Based on the details in the character description, the response should be given.

Installation

1. Installing Flask API and GPT-3 model.

   ```
   pip install flask
   pip install openai
   ```

2. Fine-tuning of GPT-3 model using **prompts** and **response** dataset

```
!openai api fine_tunes.create -t "train_dataset_prepared.jsonl" -m davinci
```

3. Connecting the endpoints of Flask API with fine-tuned GPT-3 model and Unity Engine.

   Flask API Endpoints:

```
app = Flask(__name__)


@app.route('/api', methods=['POST'])
def process_message():  # put application's code here
    message = request.form['message']
    character_name = request.form['character_name']
    response = dialog(message, character_name)
    return response


if __name__ == '__main__':
    app.run()
```

Unity Endpoints:

```
// Send the request to the API
        using (UnityWebRequest www =
UnityWebRequest.Post("http://127.0.0.1:5000/api", form))
        {
            yield return www.SendWebRequest();

            if (www.result != UnityWebRequest.Result.Success)
            {
                Debug.Log("Failed to send message to API: " + www.error);
            }
            else
            {
                // Get the response from the API
                string response = www.downloadHandler.text;
                outputArea.text = response;
            }
        }
```

Deployment

- The Flask API is deployed locally on the system to have minimum latency.
- The only API request that is sent to a distant server is to OpenAI where the fine-tuned model is requested for the response
- The Flask API will be available at http://<server_ip>:8000.
- The Flask API can also be deployed on a platform-as-a-service (PaaS) provider such as Heroku if the Gaming environment is made online.

4. Required Tools

The tools that were required to pursue the project are:

1. Ngrok
2. Telegram bot
3. Flask API
4. OpenAI GPT-3
5. Unity
6. Postman
7. PyCharm

5. Inworld

This is a developer platform that is used to create AI characters that include the creation of NPC characters for a game. In this platform, we could generate the characters from scratch which includes their physical appearance as well as their personality, traits, hobbies, voice, moods, and so on. The platform provides us with so many features for character customization and is easy to use too. Various features that are present during the character creation are:

- Core Description: We describe the character's identity and personality within this field
- Motivations: It is the goal or desire that drives the specific character
- Identity: This field includes the character's name, hobbies, stage of life, and so on
- Personality: This field includes character traits, mood, personality, and so on
- Facts and Knowledge: This field includes personal and common knowledge
- Voice: In this, we could choose the type of voice, pitch, and so on
- Dialogue Style: We could set the character's dialogue style to various types like, 'Bubbly', 'Formal', and so on
- Scenes: In this field, we add the scene in which the character is part. We create a scene by adding a detailed description of the field and then later add the characters to it.

Currently, in our project, we have created two scenes, 'Murder at the Mansion' and 'The Midnight Heist'. There are a total of six NPC characters that have been created where four characters belong to the scene 'Murder at the Mansion' and two characters to 'The Midnight Heist' scene.

5.1. Murder at the Mansion

This is a scene where John, a wealthy businessman, was murdered at his house. Four characters were developed for this scene:

- ➢ Susan [John's wife]: Susan is developed as a character who is caring and empathetic hence the way the character responds is likewise.
- ➢ Jack [John's business partner]: This character is set to be arrogant and cunning.
- ➢ Maria [Maid]: Maria is the witness of John's murder so when interrogated, the character describes what she has witnessed.
- ➢ Tom [Culprit]: This character is made to be arrogant and crooked so when interviewed, the responses are likewise, and the body language too matches his personality.

5.2. The Midnight Heist

In this scene, a convenience store in Oakville is being robbed and the scene includes two NPC characters:

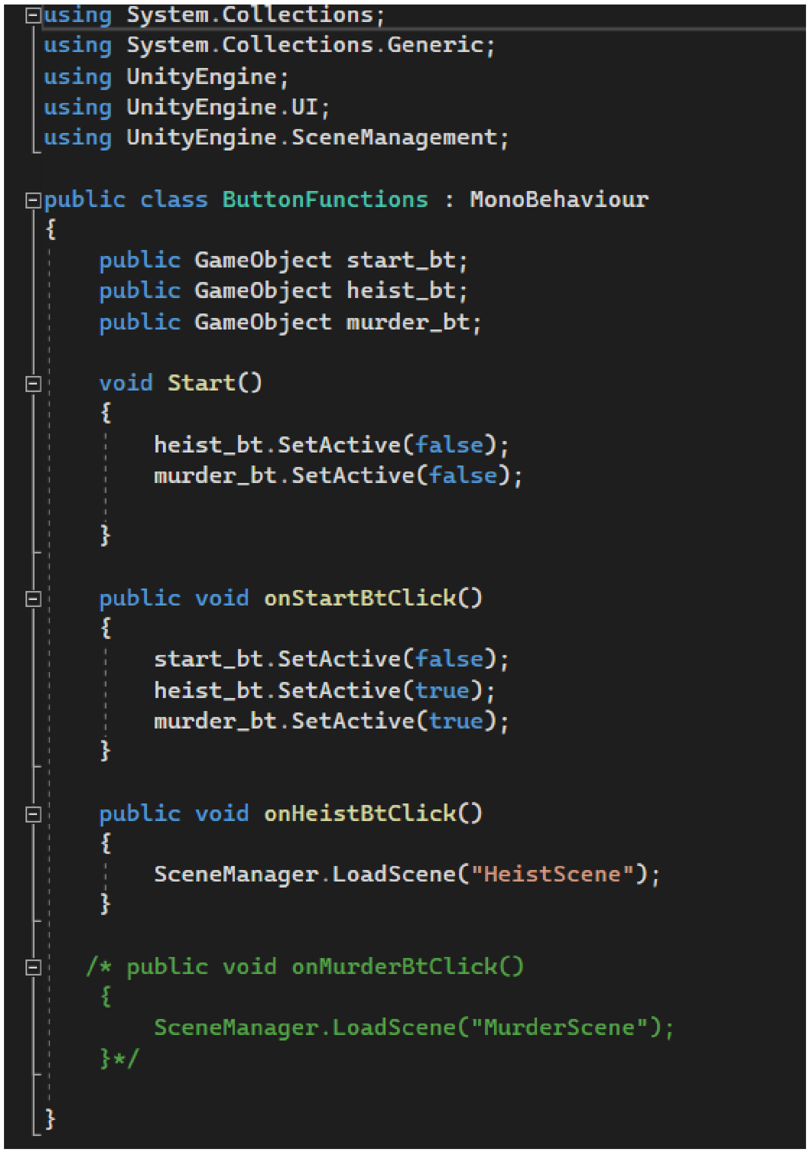
- ➢ Johnson [Owner]: This character is the owner of the store and during the robbery, the character gets injured by a gunshot in his shoulder.
- ➢ Helen [Witness]: Helen is the witness to the robbery that occurred at the store. When interrogated, the character describes what she saw and mentions Mr. Johnson's injury.

5.3. Integration of Inworld to Unity

Initially import the inworld's SDK to the unity as a custom package. Once all the packages are imported, connect our inworld account to the unity to access the avatars that we have created, this is done using the studio token that is generated within our account, the API keys are auto-populated as an option once the studio is connected to the unity.

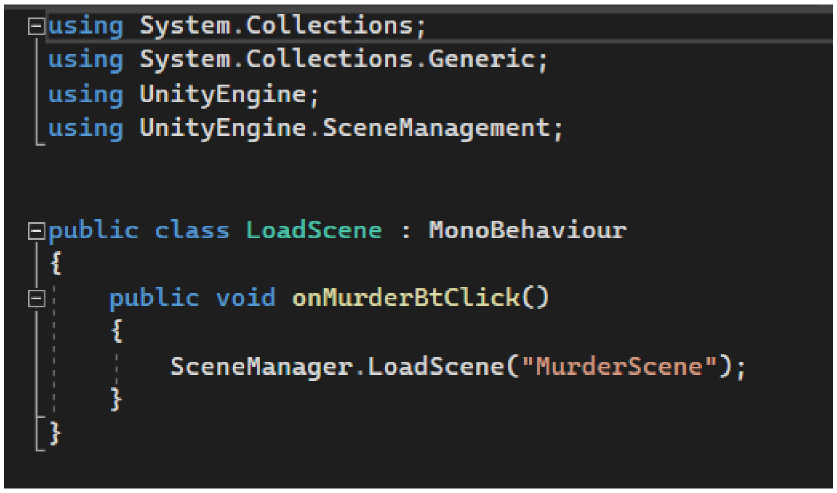
5.4. Code Integration

- Code for the initial screen's UI integration:



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class ButtonFunctions : MonoBehaviour
{
    public GameObject start_bt;
    public GameObject heist_bt;
    public GameObject murder_bt;

    void Start()
    {
        heist_bt.SetActive(false);
        murder_bt.SetActive(false);

    }

    public void onStartBtClick()
    {
        start_bt.SetActive(false);
        heist_bt.SetActive(true);
        murder_bt.SetActive(true);
    }

    public void onHeistBtClick()
    {
        SceneManager.LoadScene("HeistScene");
    }

   /* public void onMurderBtClick()
    {
        SceneManager.LoadScene("MurderScene");
    }*/

}
```

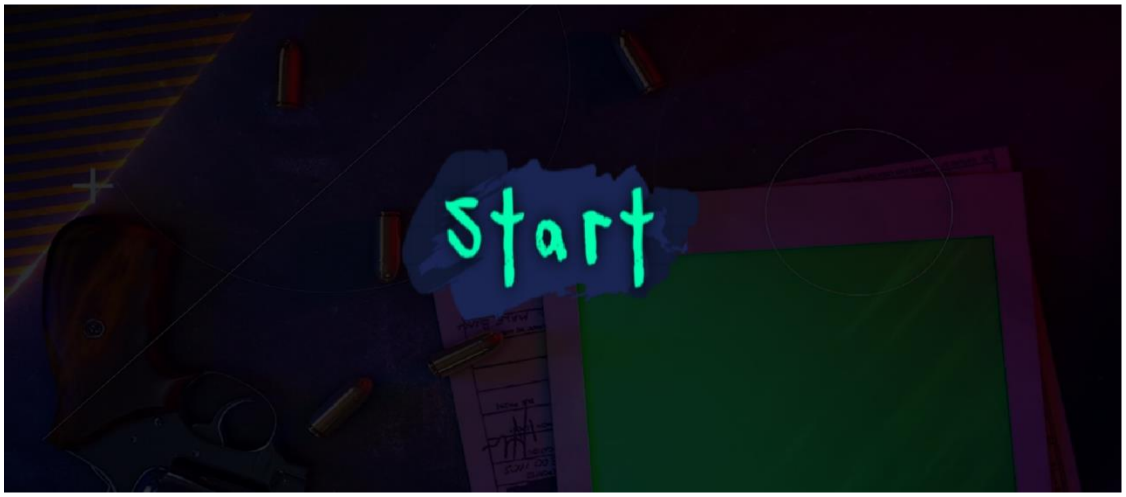
➢ Code to load scenes:



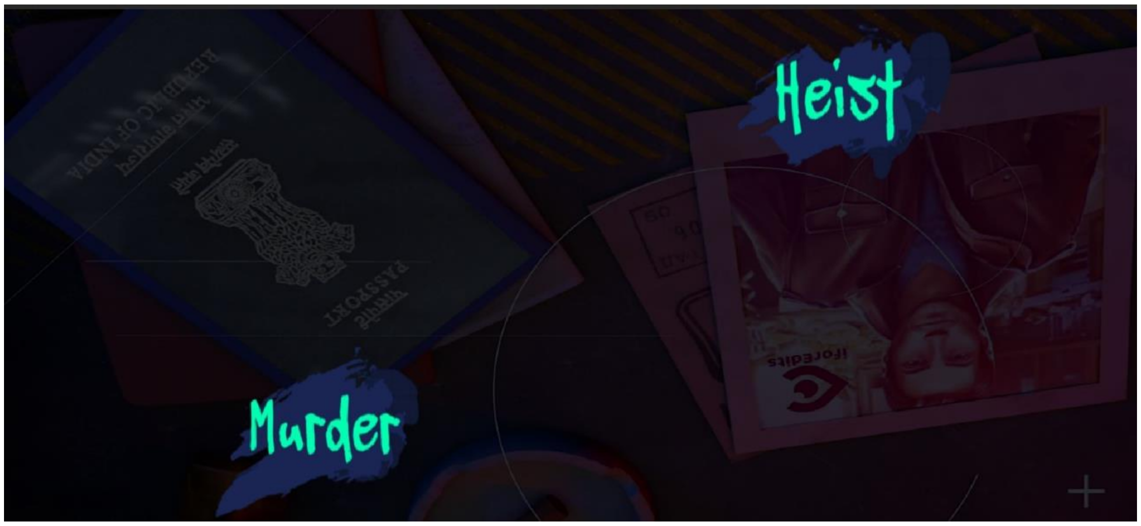
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;


public class LoadScene : MonoBehaviour
{
    public void onMurderBtClick()
    {
        SceneManager.LoadScene("MurderScene");
    }
}
```

5.5. Simulation Images

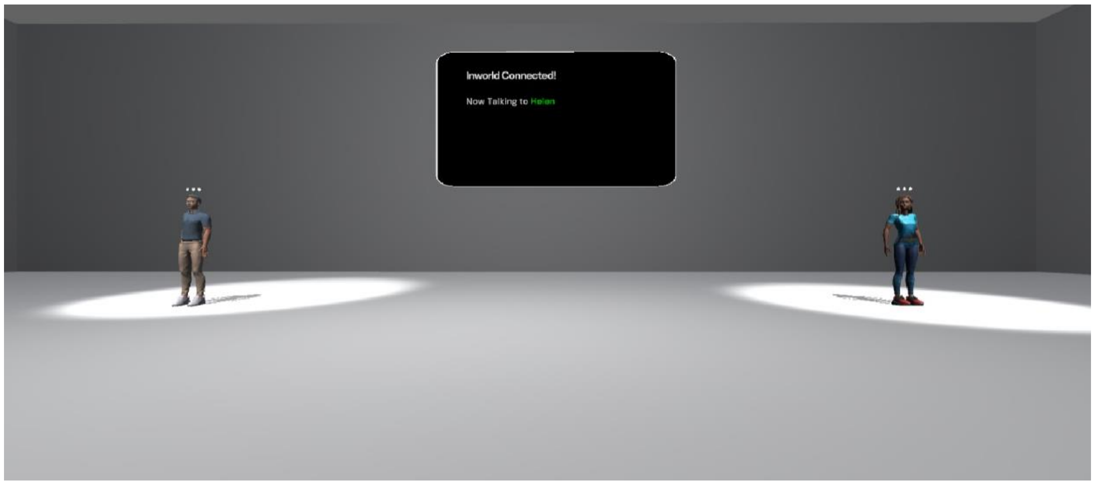
➢ Initial Screen



- Menu Screen

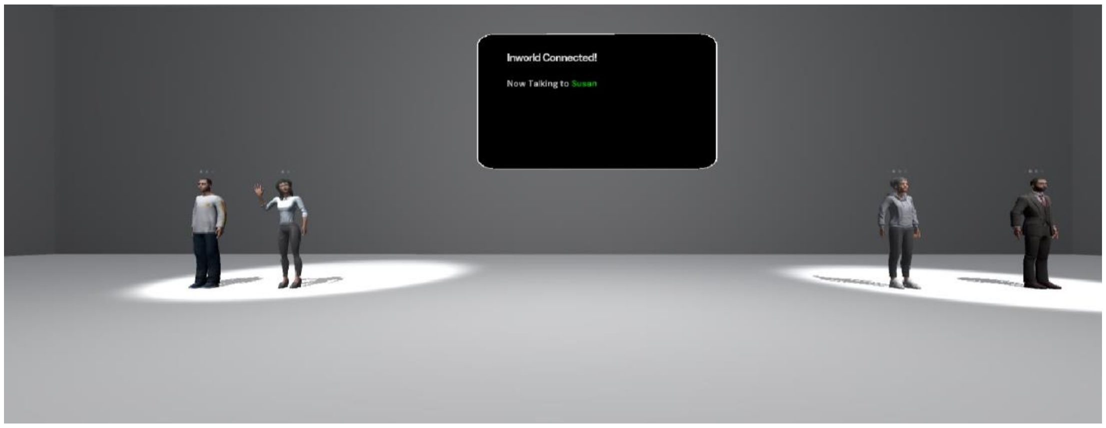


- Heist Scene:


Inworld Connected!

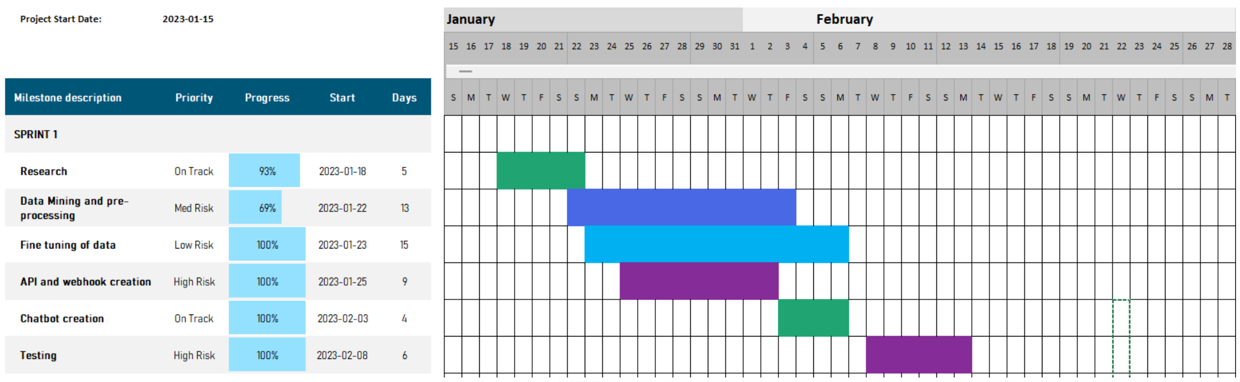
Now Talking to Helen

- Murder Scene:



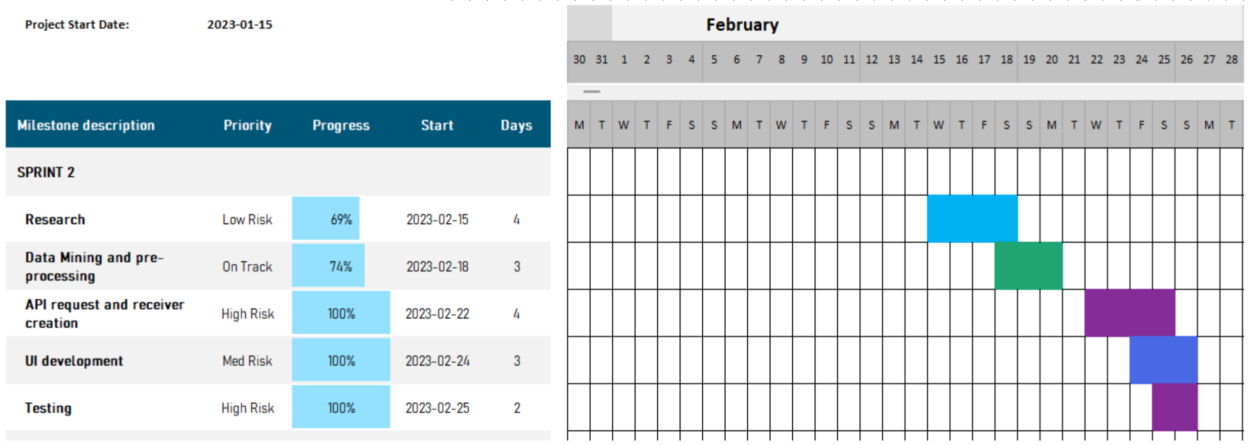
6. Gantt Chart

The project consists of five sprints which include various phases of Research, Data mining, testing, API integration, inworld integration and so on. The project marked its start on January 15th, 2023.



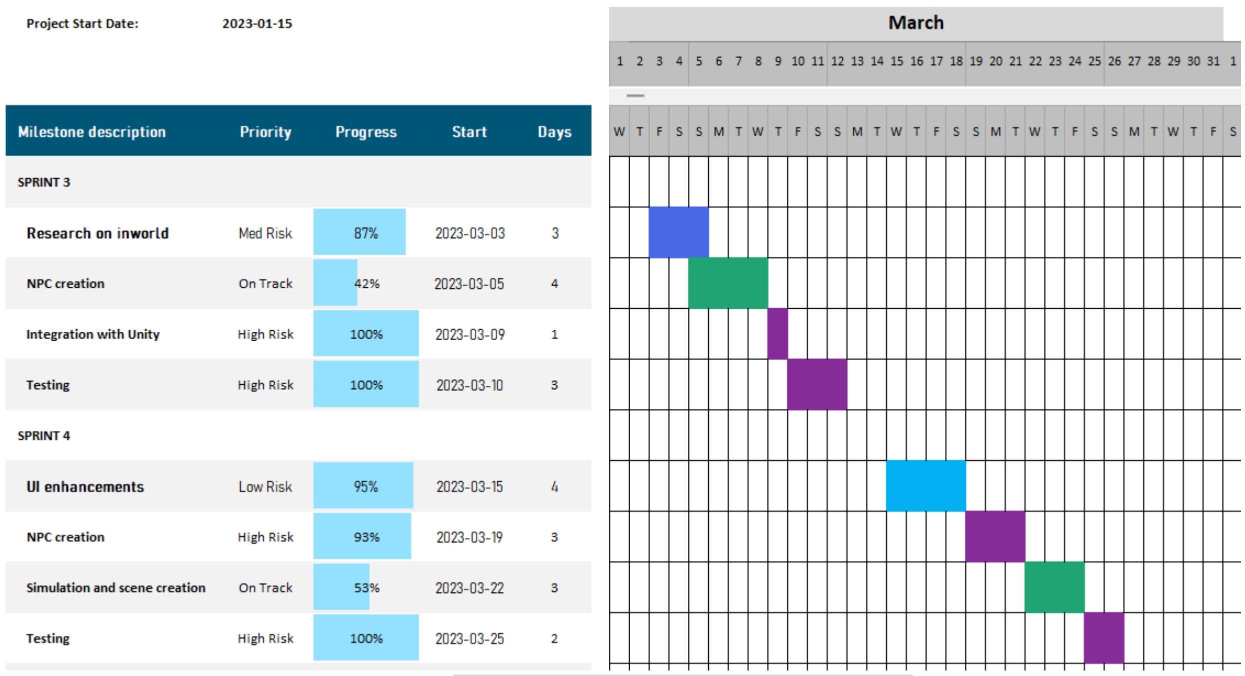
Project Start Date: 2023-01-15

Milestone description	Priority	Progress	Start	Days
SPRINT 1				
Research	On Track	93%	2023-01-18	5
Data Mining and pre-processing	Med Risk	69%	2023-01-22	13
Fine tuning of data	Low Risk	100%	2023-01-23	15
API and webhook creation	High Risk	100%	2023-01-25	9
Chatbot creation	On Track	100%	2023-02-03	4
Testing	High Risk	100%	2023-02-08	6



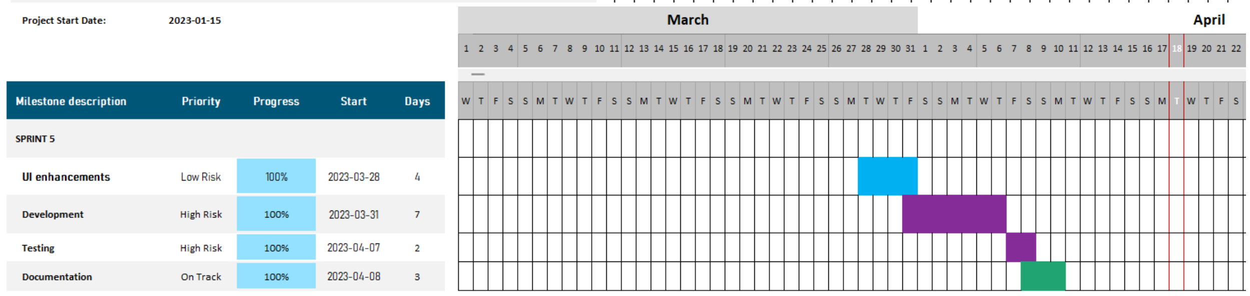
Project Start Date: 2023-01-15

Milestone description	Priority	Progress	Start	Days
SPRINT 2				
Research	Low Risk	69%	2023-02-15	4
Data Mining and pre-processing	On Track	74%	2023-02-18	3
API request and receiver creation	High Risk	100%	2023-02-22	4
UI development	Med Risk	100%	2023-02-24	3
Testing	High Risk	100%	2023-02-25	2



Project Start Date: 2023-01-15

Milestone description	Priority	Progress	Start	Days
SPRINT 3				
Research on inworld	Med Risk	87%	2023-03-03	3
NPC creation	On Track	42%	2023-03-05	4
Integration with Unity	High Risk	100%	2023-03-09	1
Testing	High Risk	100%	2023-03-10	3
SPRINT 4				
UI enhancements	Low Risk	95%	2023-03-15	4
NPC creation	High Risk	93%	2023-03-19	3
Simulation and scene creation	On Track	53%	2023-03-22	3
Testing	High Risk	100%	2023-03-25	2



Project Start Date: 2023-01-15

Milestone description	Priority	Progress	Start	Days
SPRINT 5				
UI enhancements	Low Risk	100%	2023-03-28	4
Development	High Risk	100%	2023-03-31	7
Testing	High Risk	100%	2023-04-07	2
Documentation	On Track	100%	2023-04-08	3

7. Literature Review

The creation of natural language processing (NLP) algorithms that enable virtual reality avatars to comprehend and react to spoken or written input from users in a natural and conversational way is one area of research in this discipline. That may entail training AI models on big datasets of human conversation data using methods like deep learning and machine learning in order to give them the ability to produce acceptable replies to user input.

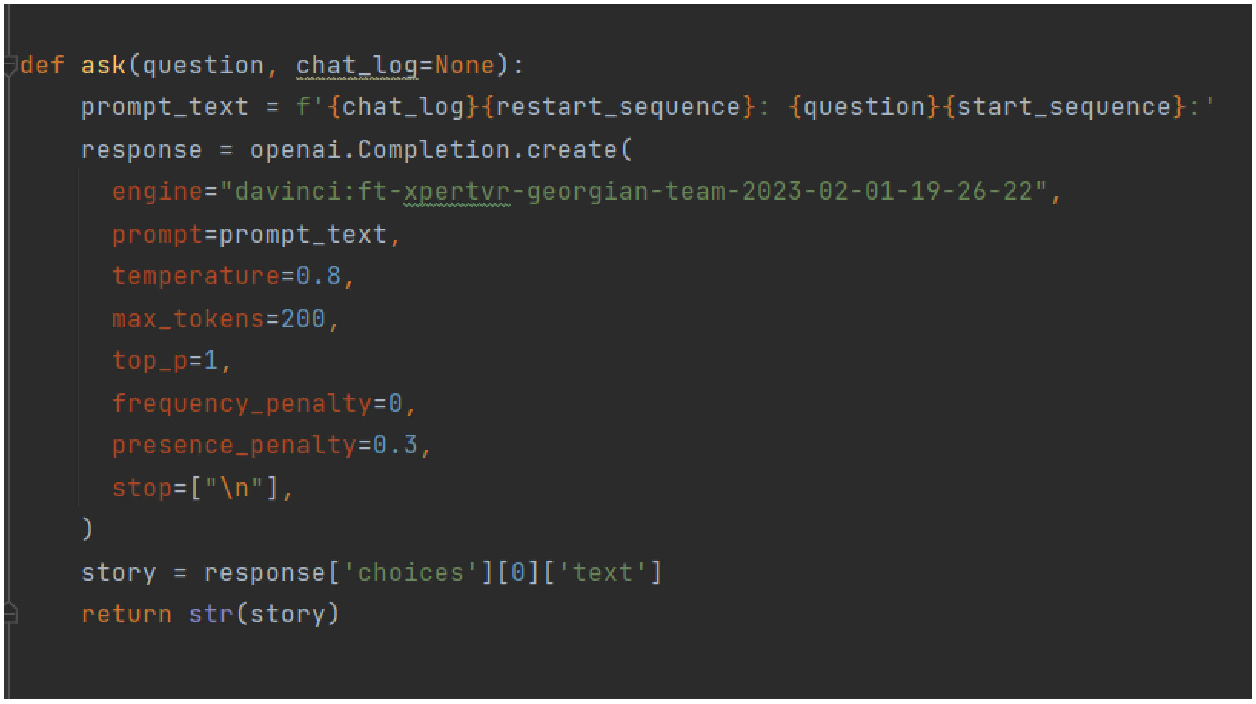
The creation and deployment of virtual reality settings that encourage natural and interesting dialogues between users and AI-powered avatars is the subject of another area of research. The creation of realistic and immersive virtual worlds using cutting-edge visuals and audio technology may be a part of this, as well as the creation of user-friendly user interfaces that enable users to effortlessly start and control discussions with AI-powered avatars.

In general, the usage of artificial intelligence (AI) in virtual reality settings for the purpose of having dialogues has the potential to completely change how we engage with technology and create new opportunities for HCI. The state of the art in this sector will probably continue to advance with additional study in this area, which will make it possible to create conversational AI systems that are even more sophisticated and natural-sounding.

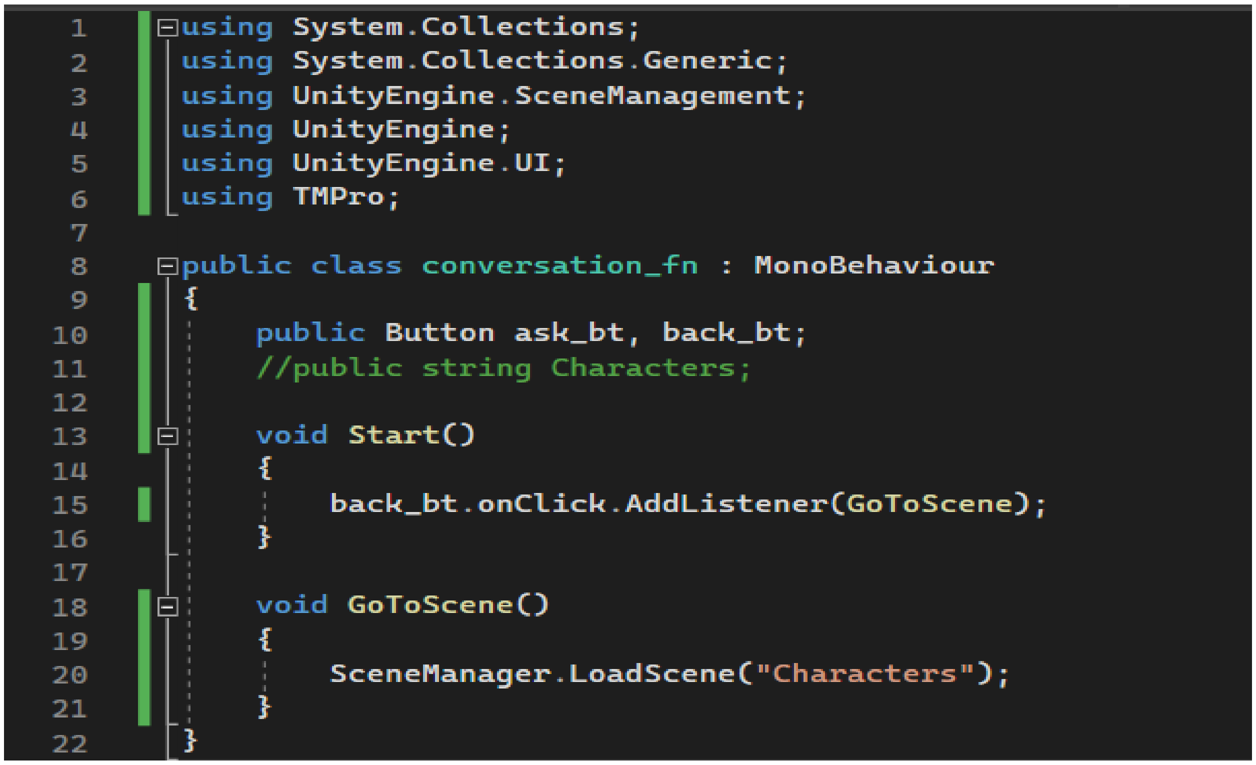
8. Deliverables with Code

The best engine, in open AI's opinion, is the DaVinci engine used in the GPT3 model. GPT3 model for Open AI performed better. So, for MVP, we are utilising this model. In the third epic, while the models are being adjusted, the model can be modified.

Generative Pretrained Transformer, also known as GPT, is a sizable language model created by OpenAI. To produce language that resembles human writing, it employs a deep learning system known as a transformer. GPT can produce content that resembles human writing in a wide range of styles and formats because it was trained on a vast amount of text data. This can be helpful for projects like content creation, text summary, and language translation.

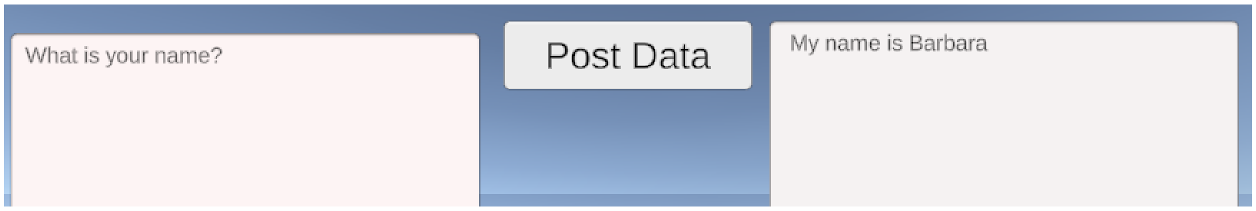


```
def ask(question, chat_log=None):
    prompt_text = f'{chat_log}{restart_sequence}: {question}{start_sequence}:'
    response = openai.Completion.create(
      engine="davinci:ft-xpertvr-georgian-team-2023-02-01-19-26-22",
      prompt=prompt_text,
      temperature=0.8,
      max_tokens=200,
      top_p=1,
      frequency_penalty=0,
      presence_penalty=0.3,
      stop=["\n"],
    )
    story = response['choices'][0]['text']
    return str(story)
```



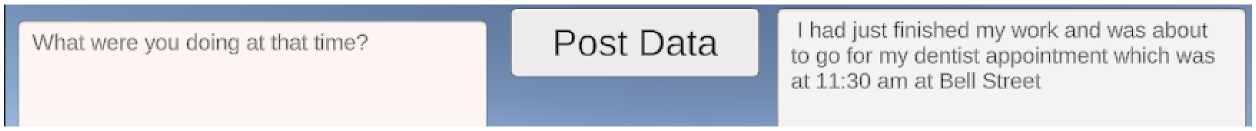
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class conversation_fn : MonoBehaviour
{
    public Button ask_bt, back_bt;
    //public string Characters;

    void Start()
    {
        back_bt.onClick.AddListener(GoToScene);
    }

    void GoToScene()
    {
        SceneManager.LoadScene("Characters");
    }
}
```

Responses of the questions from NPC model



What is your name?

My name is Barbara



What were you doing at that time?

I had just finished my work and was about to go for my dentist appointment which was at 11:30 am at Bell Street

9. Conclusion

The interface between the Unity and GPT-3 is working without interruption. The fine-tuned model is connected with Unity Game Engine via Flask API locally which has been a significant factor in reduced latency.

We are able to get accurate responses from the model representing the NPC. To improve the quality of dialogue between player character and the NPC we had to create a dialogue management system which was done during the course this project in the training dataset. The background information is the most important factor while starting a conversation with the NPC.

One important thing to note is that the memory of the NPC needs to be handled through a database to create a scalable system i.e., the NPC needs to remember the fresh conversation which would be a defining factor in future conversation it might have with the player.

Potential Improvements:

- For scaling the model to work for large number of NPCs, there should be a dialogue management system in place.
- A database for storing background information about all NPCs needs to be created.
- Improvements in code needs to be done to have improvised storing history of conversations, debugging techniques, handling incoming data about the conversations with NPCs in database, reducing latency, add more features like character positional conversation initiation, etc.
- Speech-to-text, text-to-speech aided with emotions and sentiments to be developed in the Unity's simulation environment.
- Working on behaviour patterns of NPCs.

10. References

- *OpenAI*. (n.d.). https://openai.com/
- Lankes, M., & Mirlacher, T. (2011). Affective Game Dialogues. *Lecture Notes in Computer Science*, 333–341. https://doi.org/10.1007/978-3-642-31866-5_28
- Drake, J. (2018). Planning For Non-Player Characters By Learning From Demonstration. *ResearchGate*. https://www.researchgate.net/publication/326675285_Planning_For_Non-Player_Characters_By_Learning_From_Demonstration
- Pihlgren, G. (2016). *Realistic NPCs in Video Games Using Different AI Approaches*. https://api.semanticscholar.org/CorpusID:63694745
- Buckley, D. (2023, February 19). *How to Connect to an API with Unity – Unity Apps Tutorial*. GameDev Academy. https://gamedevacademy.org/how-to-connect-to-an-api-with-unity/